Benefits and Costs of Prediction Based DVFS for NoCs at Router Level

Cristinel Ababei
Department of Electrical and Computer Engineering
Marquette University, Milwaukee WI, USA
Email: cristinel.ababei@marquette.edu

Nicholas Mastronarde
Department of Electrical Engineering
State University of New York at Buffalo, Buffalo NY, USA
Email: nmastron@buffalo.edu

***Abstract*—Power consumption remains one of the most important design objectives for network-on-chip (NoC) based systems. In this paper, we focus on the NoC component of these systems. Specifically, we introduce a new distributed dynamic voltage and frequency scaling (DVFS) algorithm that can tune the operation frequency and supply voltage of each router in the NoC dynamically in response to network load trends in order to mitigate network congestion and to reduce power consumption. The proposed distributed DVFS algorithm uses history based predictors that predict link and buffer utilizations. These predictions are used to forecast the future network load, which is used to also do proactive frequency tuning, thereby addressing potential congestion issues and reducing power consumption. When frequency throttle is used only, power consumption is reduced by up to 50% while the network latency is only slightly degraded. When frequency boost is also used, in addition to significant power reductions, network latency is also improved. We utilize the proposed DVFS algorithm as a testbed to gain new insights into the potential of DVFS for NoCs at router level.**

***Keywords*-network-on-chip; dynamic voltage and frequency scaling; power consumption; congestion; prediction**

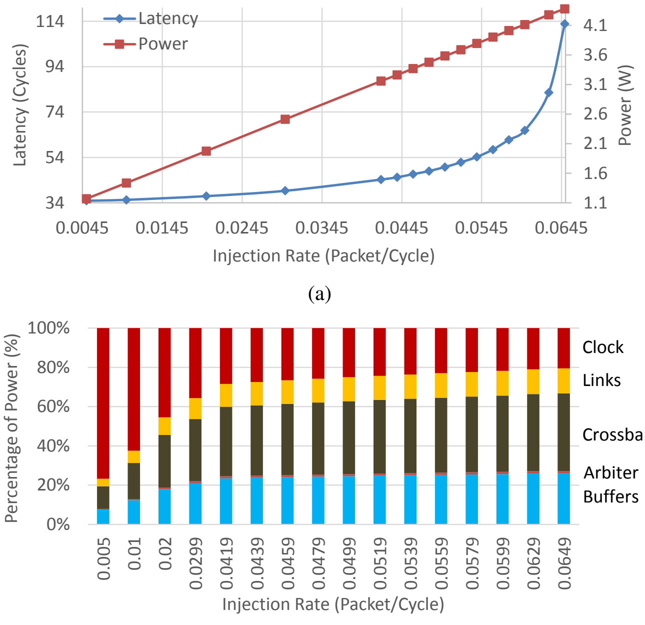
I. Introduction

One of the most effective techniques to address the problem of power consumption is dynamic voltage and frequency scaling (DVFS). Despite significant previous work, there are still open questions regarding the potential benefits of DVFS techniques at different granularity levels for NoCs. Previous studies assume that the largest contribution to the network power is the links. These studies are done for older technology nodes and report power consumption estimated with the Orion 1.0 power model, which does not include clock and link power. Other studies work with large numbers of different supply voltage and frequency levels, which may require huge area penalties. Therefore, in this paper, we revisit DVFS for NoCs in an attempt to answer some of the existing questions. Toward this goal, we introduce a new distributed DVFS algorithm for NoCs at router level, which combines some of the most promising ideas on DVFS techniques for links and routers. The proposed algorithm together with the Orion 2.0 power model are integrated within an event-driven NoC simulator, which we make publicly available.

II. The Case of DVFS for NoCs

A. The power consumed by NoCs

Power consumed by the NoC can represent up to $28-36\%$ of the total power consumed by the entire system [1], [2]. The contribution of different NoC components to this power is illustrated in Fig. 1 for an 8x8 regular mesh network exercised with uniform random traffic using our custom event-driven NoC simulator setup as described later in section V.



(a)

(b)

Fig. 1. (a) Average network latency per delivered packet and the total power consumption for an 8x8 regular mesh NoC for uniform random traffic; 65nm technology node, (b) Variation of percentages of the power consumed by router buffers, arbiter, crossbars, links, and clock components.

Fig. 1.a shows that the total power consumption increases linearly with the injection rate (i.e., network load). At small network loads, the clock component represents the majority of the power consumed by the NoC. As the network load increases, the power consumed by the router buffers and crossbar becomes dominant. At moderate network loads, the power consumed by the clock and link components is around 20% and 15%, respectively. These values are in agreement with those reported in [2] for the 80-core Intel processor prototype. These plots suggest that a comprehensive DVFS technique should address all the major components that make up the power consumed by NoCs.

B. Previous work on DVFS techniques for NoCs

Previous DVFS techniques are typically applied at either router/link level or cluster of routers level. For example, in the first category, the authors of [3] use DVFS for NoC links. They use ten different frequency levels and report 4.6X average power reduction. Adaptations of the ideas from [3] are reported in [4], [5]. Frequency boosting is used to further improve the link performance [4] while in [5] DVFS is applied to the wirelines of wireless NoCs. Frequency boosting is also used in [6]. DVFS is combined with a technique called time stealing, that allows to increase router frequency, to achieve up to 24% power savings in the power consumed by NoC routers. In the second category, many previous studies present methods to partition the NoC into several voltage-frequency islands (VFIs) and methodologies for runtime energy management [7], [8]. Because the granularity of VFIs is coarser in this case, the potential energy savings are generally less than when VF islanding is done at the router level.

Despite significant work on DVFS, it is not clear if link level DVFS is a cost effective approach and whether or not the routers should be operated at individual supply voltages and clock frequencies. For example, [3] assumes that the power consumed by links represents 82% of the total power consumed by the NoC. However, [6] assumes that the majority of the NoC power is in the buffer, crossbar, and clock related circuits of a router. The fraction of power consumed by the links is only about 13%, which is confirmed by the results on a real design [2]. The Orion 2.0 power model [9] is developed under a similar assumption. Also, many results are reported for different values of the buffer size parameter. For example, [3] works with buffer sizes of 128 flits (32-bit wide) while [6] works with buffer sizes of 4 flits (128-bit wide). Yet another parameter that varies widely in previous works is the control period, which is studied for small values of 8 to 128, and 200 cycles in [3], [4] as well as large values of 5000 cycles in [5]. In addition, many of the reported results are for network traffic that is not representative of real applications, especially for measuring and reporting latency results.

III. Background

A. Background on prediction

Prediction can be used to take preventive measures in order to mitigate or to avoid the occurrence of emergency scenarios in a *proactive* manner. For example, prediction of congestion occurrence in downstream routers can be used to trigger frequency throttling early on in upstream routers in order to lower the rate at which data is sent to downstream routers. In this paper, we use the history based prediction studied in [3] mainly due to its simplicity, which enables a cost effective *all hardware* implementation. History based prediction works with a predefined history window (HW, i.e, control period), during which the variable of interest, x, is sampled and then averaged at the end of the window. To predict the average value of the variable of interest, x_{pred}, for the next history window, the following equation is used:

$$x_{pred} = \frac{W \times x_{curr} + x_{past}}{W+1} \tag{1}$$

where, x_{curr} is the computed average value of the variable of interest in the current history window, x_{past} is the previous prediction made during the past history window, and W is a user set parameter. We use predictors for individual input-port buffer utilization (BU) and link utilization (LU) in the proposed distributed DVFS algorithm. The hardware cost of these predictors will be discussed later in section VI.

B. Background on NoC islanding

In the context of NoCs, we can find centralized or distributed DVFS techniques. Centralized techniques employ a global controller that monitors and controls the network via DVFS for each of the VFIs [7]. Distributed techniques do not depend on a global controller; instead, simpler controllers are implemented inside each router. These local controllers operate based on information that is local or gathered from the first order neighboring routers [6]. In this paper, we adopt a decentralized distributed DVFS approach because it does not require a global controller and global control signals. In addition, a distributed DVFS approach has the potential of reacting quicker to network traffic changes as it does not incur global delays and the hardware costs are smaller. Therefore, we partition the NoC into VFIs where each router represents a different island. The outgoing links are driven by router output-ports at the same VF settings as the router itself while the incoming links operate at the VF settings of neighboring routers. Fig. 2 illustrates this VF islanding.

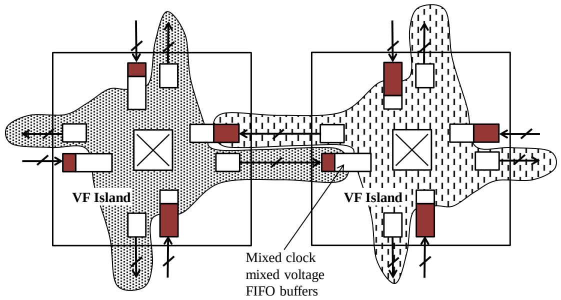


Fig. 2. Illustration of voltage frequency islanding at router level. Pattern filled areas indicate different VF islands. The links driven by a given router have the same VF settings as the router itself.

C. Background on self-similar traffic

In this paper, all simulations in section V are done for networks exercised with self-similar traffic. Self-similar traffic − which has high temporal and spatial variance, with dynamic fluctuations and bursts − is well known to better reflect the type of traffic that real applications exhibit. Our simulator integrates the self-similar traffic generator studied in [10]. Using this generator, we construct a two-level self-similar workload model similarly to [3]. At the first level, concurrent communication tasks are generated at one quarter of routers selected randomly. The *arrival* of these tasks follow a Poisson distribution with a mean of 600 cycles. The duration of these communication tasks is uniformly distributed between 600 and 1200 cycles. Within each task, at the second level of the workload model, packets are injected using the traffic generator from [10] using 128 different sources that have Pareto distributed ON/OFF periods.

IV. Proposed Distributed DVFS Algorithm at Router Level

A. *Buffer utilization as a measure of network load*

BU is a popular measure of network congestion and was used to develop DVFS algorithms for NoCs in the past. For example, the router R_2 from Fig. 3 can use information about its input buffer utilization (i.e., occupancy) to signal the neighboring routers (such as the upstream router R_1) to throttle their frequencies in cases when buffer occupancy is high. If neighboring routers could honor such throttle requests, congested routers would be able to mitigate or avoid congestion easier. In cases where buffer occupancy is moderate, the downstream router R_2 can signal the upstream router R_1 to boost the frequency, as router R_2 could handle more traffic now. Finally, when buffer occupancy is low, the downstream router R_2 can signal the upstream router R_1 to either throttle or boost the frequency. In this case, throttling could be preferred in order to save power while paying only little latency penalty because latency is anyway small at small network loads.

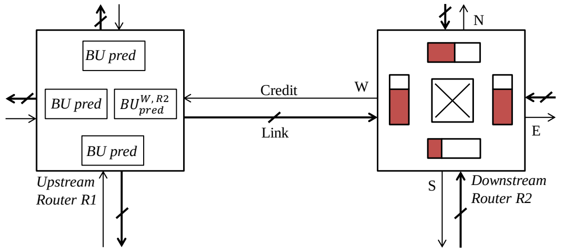


Fig. 3. Four history based BU predictors are used to compute the buffer utilizations of the four input-port buffers of downstream routers.

In our DVFS algorithm, we do not directly calculate BUs. Instead, we use history based BU predictors that are used to make DVFS decisions in a proactive fashion. Instead of computing buffer occupancy of the input-port buffers of downstream routers followed by the signaling of congestion levels to upstream routers, we implement predictors at the output-ports of the upstream routers. For example, the BU of the west input-port buffer of R_2 from Fig. 3 is predicted as $BU_{pred}^{W,R2}$ by the predictor located at the east output-port of R_1. Using predictions from all the output-ports, R_1 will decide whether to throttle or boost its frequency in the next control period as will be described later.

B. *Link utilization as a measure of network load*

In deciding whether to throttle or boost the frequency of a router, we also use information about the link utilization (LU) of the links driven by the output ports of the router because the BU alone is not a good indicator of how busy the links are. The role of LU is described with the help of Fig. 4, which shows a typical plot of the average network latency vs. the injection rate and the variations of link and buffer utilizations as functions of the injection rate. Note that LU is small for small network loads simply because few data travel through the network. For large network loads, LU is small because routers are congested and data is stalled inside buffers.

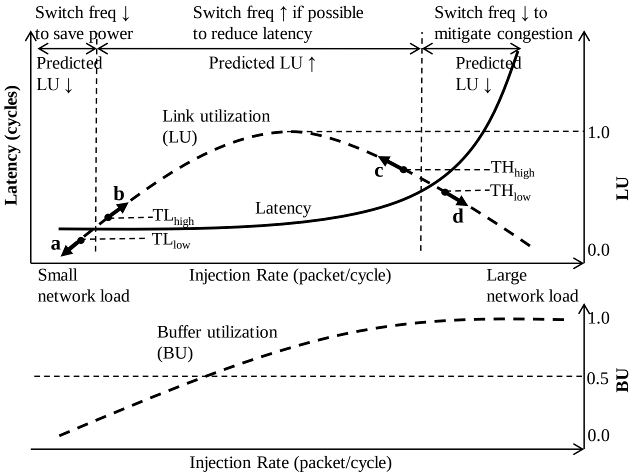


Fig. 4. Plots of typical variations of network latency, link utilization (LU) of arbitrary link, and buffer utilization (BU) of the buffer driven by link. If the predicted LU for the next control period is small, then BU can be used to distinguish between the LU at small or large network loads.

The LU is for a link connecting two routers as in Fig. 3. The BU plot is for the input buffer driven by the link. As discussed in the previous section, BU is predicted at the output-port that drives the link. The idea is to use LU prediction[1] to decide whether the frequency of the link should be throttled or boosted. For example, assume we are during the *current* control period somewhere in the middle of the LU plot and that the LU prediction we make for the *next* control period says that LU will be small. To identify the direction we move on the LU plot (indicated by the arrows **a** and **d** in Fig. 4), we also use information about the BU. When the network load is small, the BU is also small, which indicates that we are moving on the LU plot as shown by arrow **a**. Similarly, when the network load is large, the BU is also large, which indicates that we move on the LU plot in the direction of arrow **d**. Thus, by using different LU thresholds (indicated as TL_{low}, TL_{high} and TH_{low}, TH_{high} in Fig. 4), we can better control when actual frequency changes should be made.

C. *Distributed DVFS algorithm at router level*

The proposed distributed DVFS algorithm is shown in Fig. 5. It is implemented inside each router, which operates as a VFI as shown in Fig. 2, and is primarily composed of two steps that are executed at the end of each control period. First, buffer and link utilizations are computed using the predictors located at the output ports; this is done similarly to [3]. Then, the LU and BU predictions are used to decide whether to throttle or boost the router's frequency in response to the forecasted congestion in the neighboring routers; this is done similarly to [6]. Thus, the proposed algorithm can be viewed

[1] Link predictors are also located at the output ports that drive the link.

as the marriage of the DVFS for links ideas from [3] and the frequency tuning for routers ideas from [6]. In this way, both links and routers benefit from the congestion mitigation and power reduction achieved via frequency throttling and from the latency improvement achieved via frequency boosting.

```
Algorithm: Distributed DVFS for Congestion and Power Reduction
Start with each router set at $f_{base}$ and $VDD_{base}$
At end of each control period, calculate predicted BU and LU
for all input buffers of each router and the links that drive them
for $i \leftarrow 1$ to $n$ do // n: number of routers
    $counter_{switch-down} = 0$, $counter_{switch-up} = 0$
    for $j \leftarrow 1$ to $4$ do // 4: number of output ports
        $BU^j_{pred} = (W * BU^j_{curr} + BU^j_{last})/(W+1)$
        $BU^j_{last} = BU^j_{pred}$
        $LU^j_{pred} = (W * LU^j_{curr} + LU^j_{last})/(W+1)$
        $LU^j_{last} = LU^j_{pred}$
        if $BU^j_{pred} < BU_{congested}$ then // $BU_{congested} = 0.5$
            $T_{low} = TL_{low}$, $T_{high} = TL_{high}$ // 0.3, 0.4
        else
            $T_{low} = TH_{low}$, $T_{high} = TH_{high}$ // 0.6, 0.7
        end if
        if $LU^j_{pred} < T_{low}$ then
            Frequency of this link to be switched down
            $counter_{switch-down} = counter_{switch-down} + 1$
        else if $LU^j_{pred} > T_{high}$ then
            Frequency of this link to be switched up
            $counter_{switch-up} = counter_{switch-up} + 1$
        end if
    end for
    if $counter_{switch-up} > 0$ then
        Increase frequency of this router
    else if $counter_{switch-down} > 0$ then
        Decrease/throttle frequency of this router
    else
        Keep the same frequency for this router
    end if
end for
```

Fig. 5. Pseudocode of the proposed distributed DVFS algorithm. BU^j_{pred} is the predicted value of the buffer utilization of the input buffer in the downstream router, which is driven by the output port j of the currently processed router i. LU^j_{pred} is the predicted value of the link utilization of the link driven by the same output port j.

V. Simulation Results

We have implemented the proposed distributed DVFS algorithm in an event-driven NoC simulator and performed simulations for self-similar traffic (described in section III-C). In our simulations, we use an 8x8 regular mesh network whose 64 routers have input buffer size of 16 flits (64-bit wide) per virtual channel (VC), and four virtual channels. Each router has a classic four stage pipeline architecture. Routers are connected via links that are 2mm long with 64 bits bandwidth. The self-similar traffic is composed of packets with fixed packet size of 6 flits. The simulator is integrated with the Orion 2.0 power model for a 65nm technology node [9], validated with real data from the Intel's 80 core chip [2]. All simulations are done for 100000 cycles of base frequency, f_{base}, and a warmup period of 1000 cycles.

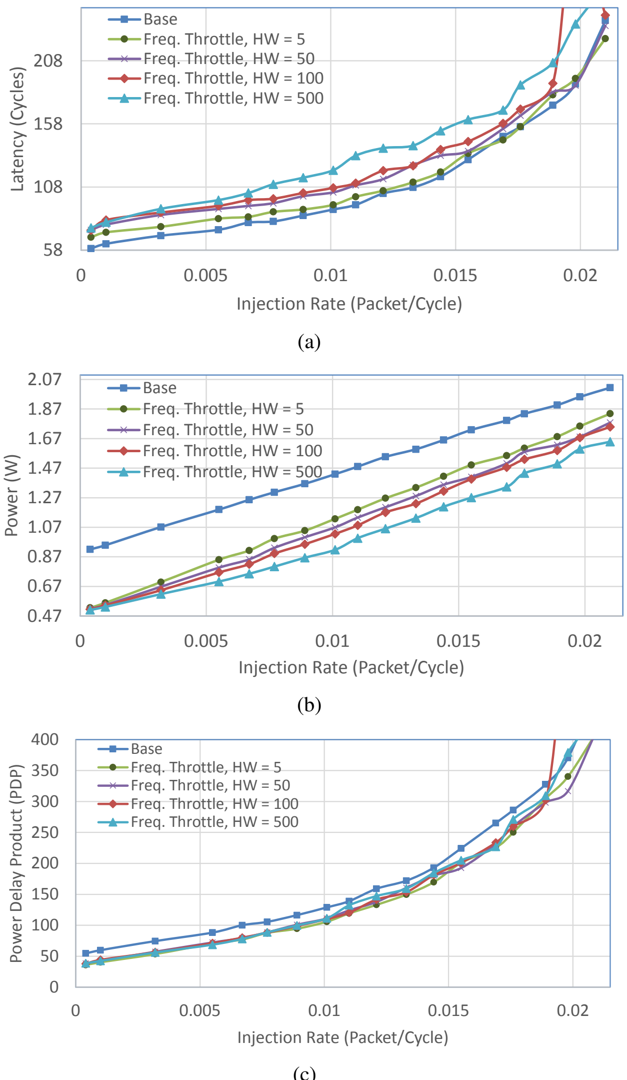
In the first part of our experiments, we use three different frequency and supply voltage values: $f_{base} = 2GHz$, $f_{throttle1} = 1.8GHz$, $f_{throttle2} = 1.6GHz$ and $VDD_{base} = 1.2V$, $VDD_{throttle1} = 1.1V$, $VDD_{throttle2} = 1.0V$. These values are in line with previous studies [6], [11]. All routers are set initially at the highest frequency, f_{base}, and supply voltage, VDD_{base}. Later, frequencies and voltages are changed dynamically using the proposed distributed DVFS algorithm from Fig. 5. The variation of the average packet latency, the total power consumption, and the power delay product[2] (PDP) are shown in Fig. 6 for four different values of the HW, which is used as the control period to make predictions for buffer and link utilizations as described in equation (1). For comparison purposes, we also show plots for the case when no DVFS algorithm is used; labeled as *Base* in Fig. 6.

We can see that when the history window is small, the power consumption is significantly reduced (Fig. 6.b) $-$ by up to 50% for small network loads $-$ and the average latency is only slightly increased (Fig. 6.a). Moreover, we see that the PDP is improved, pushing the tradeoff between latency and power consumption in a desirable direction. These results are expected, as a small HW can capture traffic variations well and therefore predictions are accurate, making the proposed algorithm an effective way to tradeoff power vs. latency. We can also see the network latency degrades with the increase in HW. That is because predictions for such long time horizons are generally wrong. More specifically, once the HW becomes comparable with the duration of tasks described in section III-C, predictions become inaccurate, which reinforces the important dependence of the quality of results on the traffic characteristics [7].

In the second part of our experiments, we change the frequency throttling approach by adding a fourth frequency level, f_{boost}. This is the frequency boost, which is 25% higher than the base or reference frequency similarly to the study in [6] and is often employed in real designs with overclocking strategies [19]. Again, all routers are set initially to the base frequency and supply voltage. Later during simulation, frequencies and voltages are changed dynamically as described in Fig. 5, but this time f_{boost} is used too.

With the addition of the frequency boost, the variation of the average packet latency, the total power consumption, and the PDP change as shown in Fig. 7. We can see that at small network loads power consumption is reduced while latency is slightly increased. That is because at small network loads there is more frequency throttling. On the other hand, at moderate and large network loads, the latency is improved significantly while the power is increased slightly. That is because, in this case, there is more frequency boosting rather than throttling and frequency throttling for small LU and large BU (right hand side of Fig. 4) helps to mitigate congestion, thereby improving overall latency. Note also that the network saturation point $-$ defined as the injection rate value where latency increases to approximately double the latency value for very small injection rates $-$ is moved in the preferred direction; that is, to the right in Fig. 7.a.

[2] We report PDP because it was shown to offer a better tradeoff between energy and delay than energy-delay or power-energy products, especially for submicron technologies [12].

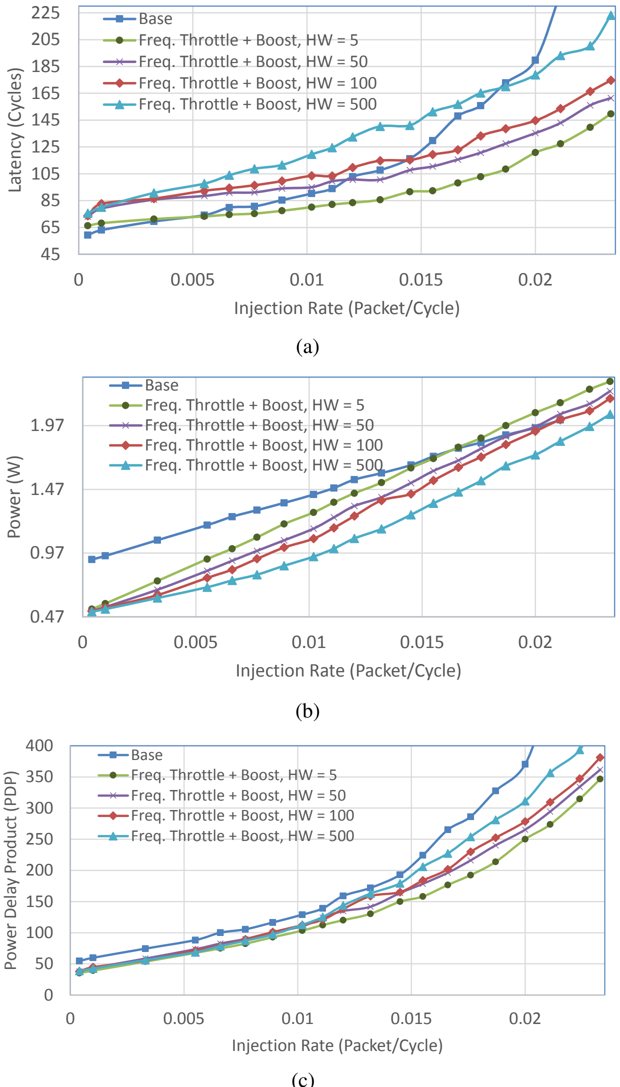


(a)

(b)

(c)

Fig. 6. (a) Average latency variation as function of injected self-similar traffic when only frequency throttle is used in the distributed DVFS algorithm from Fig. 5, (b) Power consumption, (c) Power delay product (PDP).



(a)

(b)

(c)

Fig. 7. (a) Average latency variation as function of injected self-similar traffic when both frequency throttle and frequency boost are used in the distributed DVFS algorithm from Fig. 5, (b) Power consumption, (c) Power delay product (PDP).

VI. Discussion

A. Hardware Costs

The main components required to implement the proposed distributed DVFS algorithm include:

1) Per router DC-DC multilevel converters and ring oscillators capable of generating the three or four supply voltage and frequency levels. These circuits can be implemented using the solutions studied in [13]–[15]. In the case of an 8x8 regular mesh NoC, 65nm technology node, the overhead of these circuits was estimated to be about 25% of the total area of the NoC [6]. Note that this percentage will decrease with the increase of the input buffer size. However, this area penalty and its associated power consumption penalty should not be treated as a penalty with respect to the NoC only. Most of the time, DVFS techniques are also applied to the cores and therefore these converters and ring oscillators are also used for cores, which occupy much larger areas than the NoC. Hence, this area penalty must be regarded with respect to the area of the whole system; in which case, the percentage of 25% becomes much smaller. Finally, this penalty can be reduced by applying DVFS techniques to VFIs formed by clusters of tiles, at the expense of reduced DVFS benefits.

2) Dual clock I/O buffers to facilitate asynchronous communication between routers that operate at different frequencies and supply voltages. We assume the use of mixed-clock mixed-voltage buffer designs from [16]–[18] because they have been reported as having negligible area and power penalty [6], [7].

3) The history predictors for calculating the buffer and link utilizations together with the logic to implement the algorithm from Fig. 5. Each router has four BU predictors and four LU predictors. We assume that these predictors are implemented using *all hardware* circuit solutions similar to those studied in [3]. Previous studies reported that the overhead of such predictor circuits as well as of the remaining control logic to implement algorithms whose complexity is similar to that of

the proposed DVFS algorithm is also negligible [3], [4], [6].

B. Cons of doing DVFS at router level

1) The hardware penalty discussed in the previous section represents one of the main costs. Whether this cost is worthwhile, will probably depend on the application specific domain.

2) DVFS techniques adjust the frequency and voltage in specific order. When the frequency is adjusted from high to low, it is scaled down before the voltage is decreased. When the frequency is adjusted from low to high, the voltage is increased before the frequency is scaled up. Either of these activities requires switching time, which we must pay as penalty and can be anywhere from one cycle up to 50 or more cycles of base frequency [3], [5], [14], depending on the actual DC-DC converter circuits and technology node. We find the need for fast switching DC-DC converters as the number one challenge in realizing the potential of prediction based DVFS because only small switching times would allow us to work with relatively short control periods, when predictions are accurate.

3) In our experiments, we found that results are sensitive to the *hardcoded* thresholds utilized in the proposed distributed DVFS algorithm. Previous studies also use similar hardcoded thresholds. For example, the thresholds 0.3 and 0.4 from Fig. 5 have a significant impact over the tradeoff between latency and power at small network loads. If these thresholds are selected too small, then, frequency boosting will be triggered very early and the network ends up operating at high frequency most of the time. On the other hand, selecting the thresholds 0.6 and 0.7 from Fig. 5 too large may delay the time when frequency is throttled, which may be too late to have a satisfactory impact on mitigating congestion, thereby improving also latency. Selection of these thresholds can represent a design difficulty, especially when DVFS algorithms are implemented all in hardware. Nevertheless, this con is also a pro because the hardcoded thresholds can be used to control/tune[3] the tradeoff between power savings and latency penalty described in Fig. 6 and Fig. 7.

C. Pros of doing DVFS at router level

1) DVFS techniques at router level can be utilized to significantly reduce power consumption of NoCs by trading-off acceptable latency degradation.

2) We find that improving the power delay product (PDP) is one of the most important benefits of the proposed DVFS algorithm.

VII. Conclusion

We propose a distributed DVFS algorithm for NoCs whose objective is both to mitigate NoC congestion and to reduce power consumption via online distributed voltage and frequency tuning of each router individually. Using the proposed algorithm as a testbed, we investigate the potential benefits and limitations of distributed DVFS techniques for NoCs at router level. We find that while DVFS offers a tunable tradeoff between latency and power and can be used to improve the power delay product, it poses the challenges of fast switching DC-DC converters and of hardware overheads. Our entire simulation framework is publicly available at [20].

Acknowledgment

This work was supported in part by the Dept. of Electrical and Computer Engr. at Marquette University. We thank Susan Schneider for feedback on an early draft of this presentation.

References

[1] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, “A 5-GHz mesh interconnect for a teraïňĆops processor," *IEEE Micro*, vol. 27, no. 5, pp. 51-61, Sep.-Oct. 2007.

[2] S.R. Vangal et al., “An 80-tile sub-100-W TeraFLOPS processor in 65-nm CMOS," *IEEE Journal of Solid-state Circuits*, vol. 43, no. 1, pp. 29-41, Feb. 2008.

[3] L. Shang, L.-S. Peh and N. K. Jha, “Dynamic voltage scaling with links for power optimization of interconnection networks," *HPCA*, 2003.

[4] S.E. Lee and N. Bagherzadeh, “A variable frequency link for a power-aware network-on-chip (NoC)," *Integration*, vol. 42, no. 4, pp. 479-485, 2009.

[5] J. Murray, P.P. Pande, and B. Shirazi, “DVFS-enabled sustainable wireless NoC architecture," *SOCC*, 2012.

[6] A.K. Mishra, A. Yanamandra, R. Das, S. Eachempati, R.R. Iyer, N. Vijaykrishnan, and C.R. Das, “RAFT: a router architecture with frequency tuning for on-chip networks," *J. Parallel Distrib. Comput.*, vol. 71, no. 5, pp. 625-640, 2011.

[7] U.Y. Ogras, R. Marculescu, D. Marculescu, and E.G. Jung, “Design and management of voltage-frequency island partitioned Networks-on-Chip," *IEEE Trans. on VLSI Syst.*, vol. 17, no. 3, pp. 330-341, 2009.

[8] D. Mirzoyan, S. Stuijk, B. Akesson, and K. Goossens, “Throughput analysis and voltage-frequency island partitioning for streaming applications under process variation," *ESTImedia*, 2013.

[9] A.B. Kahng et al., “ORION 2.0: A power-area simulator for interconnection networks," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 20. no. 1, pp. 191-196, Jan. 2012.

[10] Glen Kramer, Synthetic self-similar traffic generation, 2014. [Online]. Available: http://glenkramer.com/ucdavis/trf_research.html.

[11] A.K. Coskun, T.S. Rosing, and K.C. Gross, “Utilizing predictors for efficient thermal management in multiprocessor SoCs," *IEEE Trans. on CAD of Integrated Circuits and Systems (TCAD)*, vol. 28, no. 10, pp. 1503-1516, 2009.

[12] D. Sengupta and R.A. Saleh, “Generalized power-delay metrics in deep submicron CMOS designs," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 26, no. 1, pp. 183-189, Dec. 2007.

[13] W. Kim, D. Brooks, G.-Y. Wei, “A fully-integrated 3-level DC-DC converter for nanosecond-scale DVFS," *IEEE J. Solid-State Circuits*, vol. 47, no. 1, pp. 206-219, 2012.

[14] S. Sheikhaei, M. Alimadadi, G.G.M. Lemieux, S. Mirabbasi, W.G. Dunford, and P.R. Palmer, “Energy recycling from multigigahertz clocks using fully integrated switching converters," *IEEE Trans. on Power Electronics*, vol. 28. no. 9, pp. 4227-4239, 2013.

[15] K. Hausman, G. Gaudenzi, J. Mosley, and S. Tempest, US Patent 4978927 - Programmable Voltage Controlled Ring Oscillator, 1990.

[16] E.-J. Kim et al., “A holistic approach to designing energy-efficient cluster interconnects," *IEEE Trans. on Computers*, vol. 54, pp. 660-671, June 2005.

[17] T. Chelcea and S.M. Nowick, “A low latency FIFO for mixed-clock systems," *IEEE Comput. Soc. Workshop VLSI*, 2000.

[18] A.-M. Rahmani, P. Liljeberg, J. Plosila, and H. Tenhunen, “Design and implementation of reconfigurable FIFOs for voltage/frequency island-based networks-on-chip," *Microprocessors and Microsystems*, vol. 37, no. 4-5, pp. 432-444, 2013.

[19] D. Lo and C. Kozyrakis, “Dynamic management of TurboMode in modern multi-core chips," *HPCA*, 2014.

[20] Software downloads at MES Lab, 2014. [Online]. Available: http://dejazzer.com/software.html.

[3] This can be achieved, for example, by storing several different threshold values and then using them dynamically during runtime.